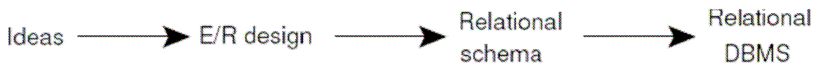
The Relational Model

CS 317, Fall 2007

Course outline

- Weeks 1–7: Relational Data Models
  - E/R models. Weeks 1–3.
  - The relational model. Next 4 weeks.
    - Convert E/R to relational schemas.
    - Functional and multi valued dependencies.
    - "Normalize" relations to improve a relational design.



Relational Model

- Built around a single concept for modeling data: *the relation or table*.
- Supports high-level programming language (SQL).
- Has an elegant mathematical design theory.
- Most current DBMS are relational.

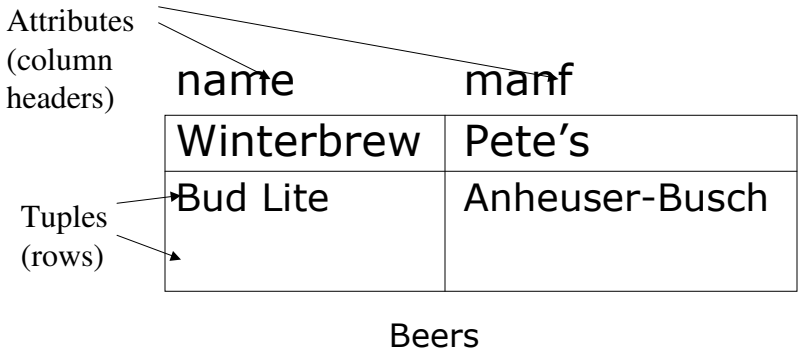
The Relation

- A relation is a two-dimensional table:
  - Relation = table.
  - Attribute = column name.
  - Tuple = row (not the header row).
  - Database = collection of relations.

CoursesTaken

Student	*Course*	*Grade*
Hermione Grainger	Potions	A-
Draco Malfoy	Potions	B
Harry Potter	Potions	A
Ron Weasley	Potions	C

Example 2



Beers

The Schema

- The schema of a relation is the name of the relation followed by a parenthesized list of attributes.
  - **`CoursesTaken(Student, Course, Grade)`**
  - Example: **`Beers(name, manf)`** or **`Beers(name: string, manf: string)`**
- A design in a relational model consists of a set of schemas.
  - Such a set of schemas is called a relational database schema.

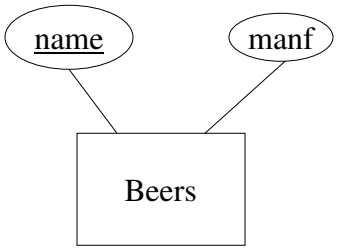
Equivalent Representation of a Relation

- A relation is a set of tuples and not a list of tuples.
  - Order in which we present the tuples does not matter.
- The attributes in a schema are a set (not a list).
  - Schema is the same irrespective of order of attributes.
  - We specify a "standard" order when we introduce a schema.
  - If we reorder attributes, we must also reorder tuples.
- How many equivalent representations are there for a relation with `m` attributes and `n` tuples? (M! N!)

Going from ER to Relational

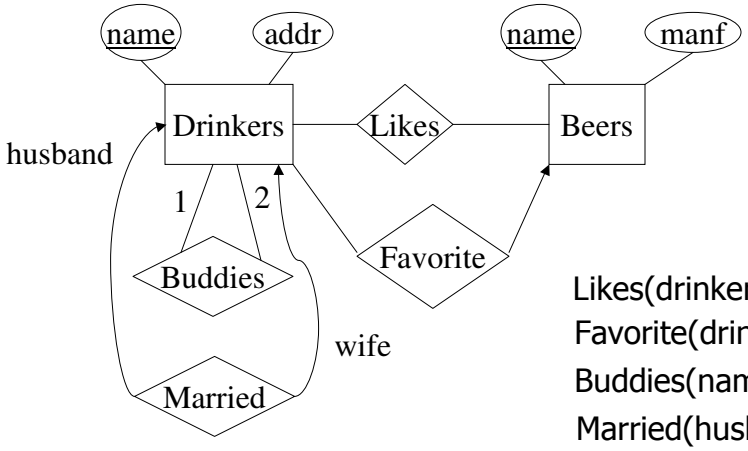
- Entity set -> relation.
  - Attribute of an entity set -> attribute of a relation.
- Relationship -> relation whose attributes are
  - Attribute of the relationship itself.
  - Key attributes of the connected entity sets.
- Several special cases:
  - Weak entity sets.
  - Combining relations (especially for many-one relationships).
  - Is-a relationships and subclasses.

Entity Set -> Relation



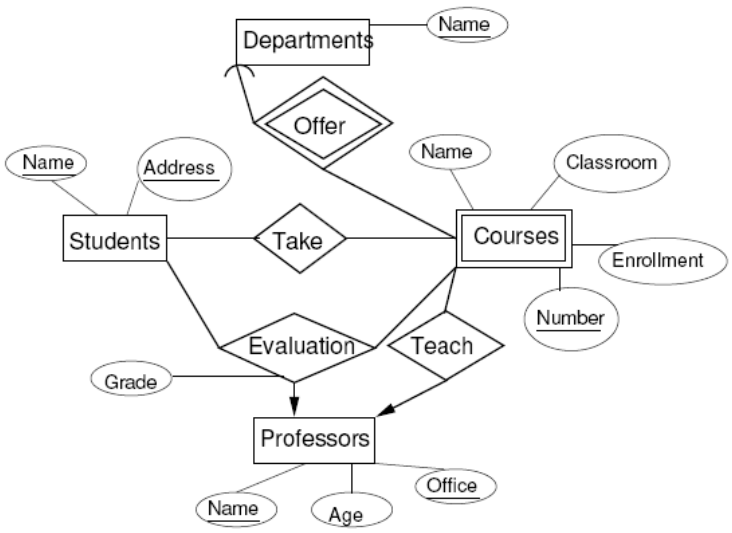
Relation: **`Beers(name, manf)`**

Relationship -> Relation



Likes(drinker, beer)
Favorite(drinker, beer)
Buddies(name1, name2)
Married(husband, wife)

Complete Example



Schemas for Non Weak entity sets

- For each entity set, create a relation with the same name and with the same set of attributes.

  `Students(Name, Address)`

  `Professors(Name, Office, Age)`

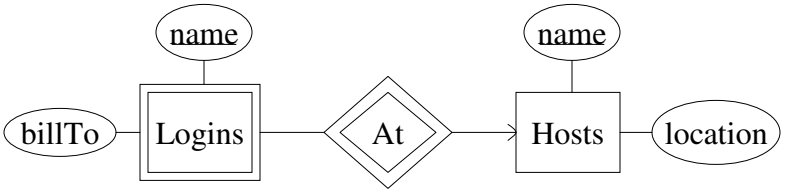
  `Departments(Name)`

Weak Entity Sets

- For each weak entity set `W`, create a relation with the same name whose attributes are
  - Attributes of `W` &
  - Key attributes of the other entity sets that help form the key for `W`.

`Courses(Number, DepartmentName, CourseName, Classroom, Enrollment)`

Another Example



Hosts(hostName, location)
Logins(loginName, hostName, billTo)
~~At(loginName, hostName, hostName2)~~

At becomes part of Logins

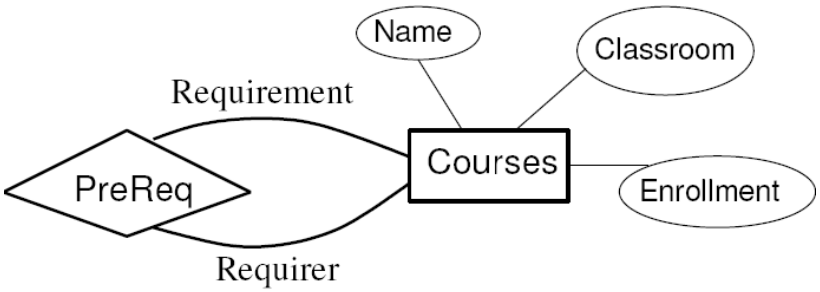
Must be the same

Schemas for Non supporting Relationships

- For each relationship, create a relation with the same name whose attributes are
  - Attributes of the relationship itself.
  - Key attributes of the connected entity sets (even if they are weak).

```
Take(StudentName, Address, Number,
  DepartmentName)

Teach(ProfessorName, Office, Number,
  DepartmentName)

Evaluation(StudentName, Address,
     ProfessorName, Office, Number,
     DepartmentName)
```

Roles in Relationships



- If an entity set E appears k > 1 times in a relationship R (in different roles), the key attributes for E appear k times in the relation for R, appropriately renamed.

```
PreReq(  RequirerNumber,
         RequirerDeptName,
         RequirementNumber,
         RequirementDeptName)
```

Combining Relations

- Consider many-one Teach relationship from Courses to Professors.
- Schemas are:

```
Courses(Number, DepartmentName,
     CourseName, Classroom, Enrollment)

Professors(Name, Office, Age)

Teach(Number, DepartmentName,
     ProfessorName, Office)
```

- The key for `Courses` uniquely determines all attributes of `Teach`.
- We can combine the relations for `Courses` and `Teach` into a single relation whose attributes are
  - All the attributes for `Courses`,
  - Any attributes of `Teach`, and
  - The key attributes of `Professors`.

Rules for combining Relationships

- We can combine into one relation Q
  - The relation for an entity set E and
  - all many-to-one relations R1, R2, . . . Rk from E to other entity sets E1, E2, . . . , Ek , respectively.
- The attributes of Q are
  - all the attributes of E,
  - any attributes of R1, R2, . . . Rk , and
  - the key attributes of E1, E2, . . . , Ek .
- Can we combine E and R if R is a many-many relationship from E to F?

Combining Example

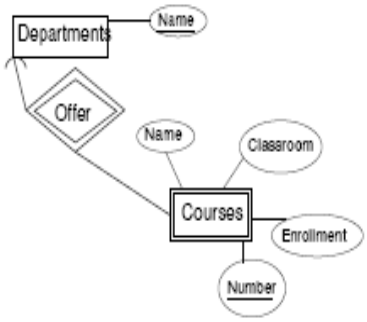
`Drinkers(name, addr)` and **`Favorite(drinker, beer)`**

combine to make

Drinker1(name, addr, favBeer)

Supporting Relationships

- Schema for Departments is Departments(Name).
- Schema for Courses is:
  - **`Courses(Number, DepartmentName, CourseName, Classroom, Enrollment)`**
- What is the schema for Offer?
  - **`Offer(Name, Number, DepartmentName)`**
  - But **`Name`** and **`DepartmentName`** are identical, so the schema for Offer is **`Offer(Number, DepartmentName)`.**



The schema for Offer is a subset of the schema for the weak entity set, so we can dispense with the relation for Offer.

Summary of Weak Entity Sets

- If W is a weak entity set, the relation for W has a schema whose attributes are
  - all attributes of W,
  - all attributes of supporting relationships for W, and
  - for each supporting relationships for W to an entity set E, the key attributes of E.
- There is no relation for any supporting relationship for W.

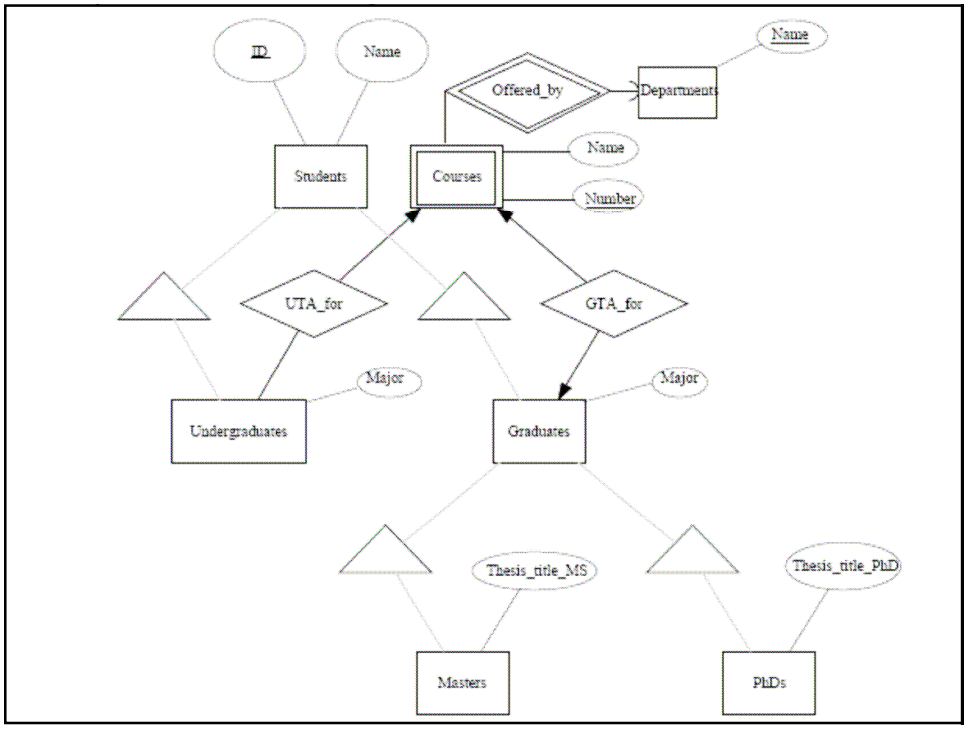
Is-A to Relational

- Three approaches:

1. E/R viewpoint
2. Object-oriented viewpoint
3. "Flatten" viewpoint

Rules of a Is-A Hierarchy

- The hierarchy has a root entity set.
- The root entity set has a key that identifies every entity represented by the hierarchy.
- A particular entity can have components that belong to entity sets of any subtree of the hierarchy, as long as that subtree includes the root.



E/R Approach

- Create a relation for each entity set.
- The attributes of the relation for a non-root entity set E are
  - the attributes forming the key (obtained from the root) and
  - any attributes of E itself.
- An entity with components in multiple entity sets has tuples in all the relations corresponding to these entity sets.
- Do not create a relation for any is-a relationship.
- Create a relation for every other relationship.

Applied to our example

```
Students(ID, Name)
Undergraduates(ID, Major)
Graduates(ID, Major)
Masters(ID, Thesis_title_MS)
PhDs(ID, Thesis_title_PhD)
UTA_for(ID, CourseNumber, DepartmentName)
GTA_for(ID, CourseNumber, DepartmentName)
```

"Flattening Approach"

- Create a single relation for the entire hierarchy.
- Attributes are
  - the key attributes of the root and
  - the attributes of each entity set in the hierarchy.
- Handle relationships as before.
- Entities have NULL in attributes that don't belong to them

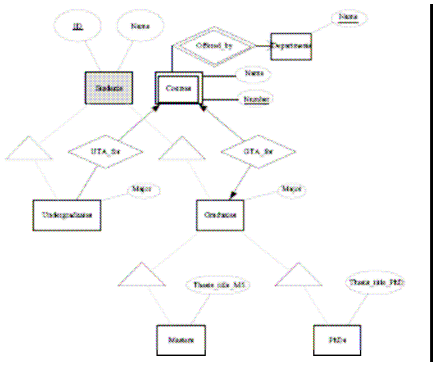
```
Students(ID, Name, UGMajor, GMajor,
        Thesis_title_MS, Thesis_title_PhD).
```

OO Approach

- Treat entities as objects belonging to a single class.
- "Class" sub tree of the hierarchy that includes the root.
- Enumerate all sub trees of the hierarchy that contain the root.
- For each such sub tree,
  - Create a relation that represents entities that have components in exactly that sub tree.
  - The schema for this relation has all the attributes of all the entity sets in that sub tree.
- Schema of the relation for a relationship has key attributes of the connected entity sets.

Subtrees

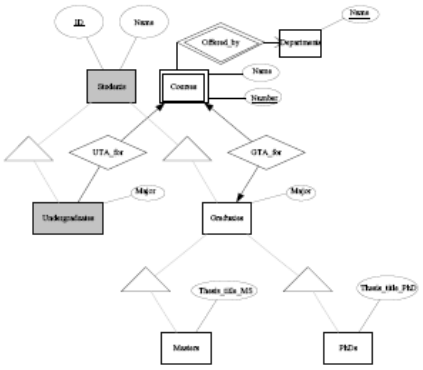
`Students (ID)`



Subtrees

Students (ID)

StudentsUGs(ID, Major)

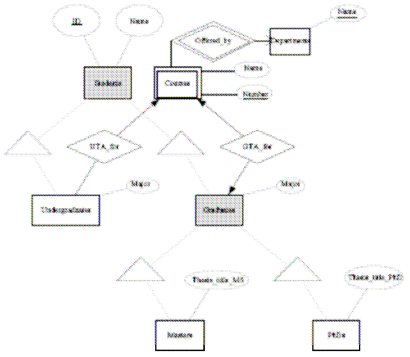


Subtrees

Students (ID)

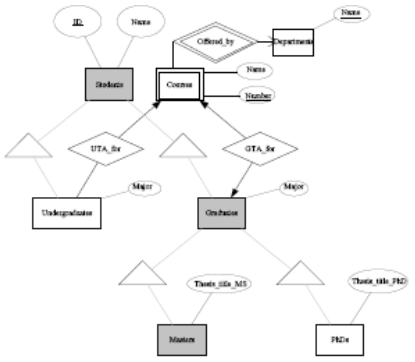
StudentsUGs(ID,Major)

StudentsGs(ID,Major)



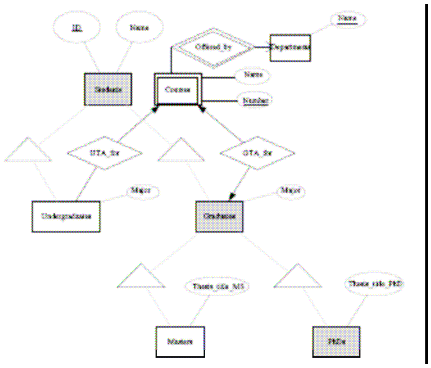
Subtrees

```
Students (ID)
StudentsUGs(ID,Major)
StudentsGs(ID,Major)

StudentsGsMasters(
            ID,
            Major,
      Thesis_title_MS)
```



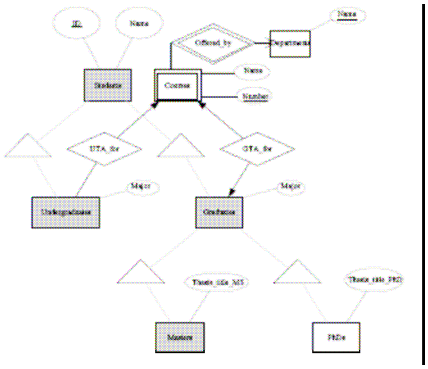
Subtrees

```
Students (ID)
StudentsUGs(ID,Major)
StudentsGs(ID,Major)
StudentsGsMasters(ID,Major
   ,    Thesis_title_MS)

StudentsGsPhDs(ID, Major,
       Thesis_title_PhD)
```

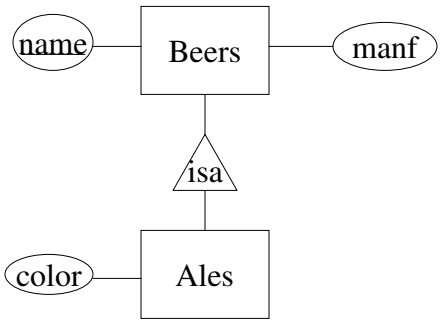


Subtrees

```
Students (ID)
StudentsUGs(ID,Major)
StudentsGs(ID,Major)
StudentsGsMasters(ID,Major
   ,   Thesis_title_MS)
StudentsGsPhDs(ID, Major,
       Thesis_title_PhD)

StudentsUGsGsMasters(ID,
       UGMinor, GradMinor,
       Thesis_title_MS)
```



Example



Object-Oriented

name	manf
Bud	Anheuser-Busch

Beers

name	manf	color
Summerbrew	Pete's	dark

Ales

Good for queries like "find the color of ales made by Pete's."

E/R Style

name	manf
Bud	Anheuser-Busch
Summerbrew	Pete's

Beers

name	color
Summerbrew	dark

Ales

Good for queries like "find all beers (including ales) made by Pete's."

Flattening

name	manf	color
Bud	Anheuser-Busch	NULL
Summerbrew	Pete's	dark

Beers

Saves space unless there are *lots* of attributes that are usually NULL.

Comparison of the approaches

- *Answering queries*
  - It is expensive to answer queries involving several relations.
  - Queries about Students in general.
  - Queries about a particular subclass of Students
- *Number of relations for n entities in the hierarchy.*
  - We like to have a small number of relations.
  - Flatten: 1.
  - E/R: n.
  - OO: can be 2^n.
- *Redundancy and space usage*
  - OO: Only one tuple per entity.
  - Flatten: May have a large number of NULLs.
  - E/R: Several tuples per entity, but only key attributes repeated.